



# Introduction CORBA

Alexandre Denis – [Alexandre.Denis@inria.fr](mailto:Alexandre.Denis@inria.fr)

**Inria Bordeaux – Sud-Ouest  
France**

# Contexte

- Besoins
  - Interconnexion de systèmes d'information
  - Réutilisation de codes existants
- Hétérogénéité -> interopérabilité
  - Matériel, langage, OS, etc.
- Un **standard** d'architecture d'applications réparties
  - Intéropérabilité entre plate-formes, langages, etc.

# OMG – Object Management Group

- Consortium à but non-lucratif fondé en 1989
  - Plus de 850 membres (constructeurs, SSI, utilisateurs, recherche)
- **Standards** pour les applications réparties
- Fonctionnement
  - Propositions, discussions, vote
  - -> **spécifications**
- Ne fournit pas d'implémentation
- ex.: OMA, CORBA, UML, MOF

# Vue d'ensemble de CORBA

- CORBA - Common Object Request Broker Architecture
  - CORBA 1.0 – 1991, modèle objet
  - CORBA 2.0 – 1995, interopérabilité, IIOP
  - CORBA 3.0 – 2002, modèle composant
- Paradigme unifié : appel à des objets distants
- Langage d'interface unifié : IDL
- Large collection de services communs
  - Présentés sous forme d'objets CORBA

# Propriétés de CORBA

- Intergiciel suivant un standard ouvert
  - Plusieurs implémentations
  - ex.: omniORB, MICO, TAO, ORBacus, JacORB, ORBexpress, ORBit, VBOrb, R2CORBA, IIOP.NET, ...
- **Interopérabilité**
  - Entre langages
  - Entre machines, OS
  - Entre constructeurs

# ORB

- ORB – bus logiciel entre applications

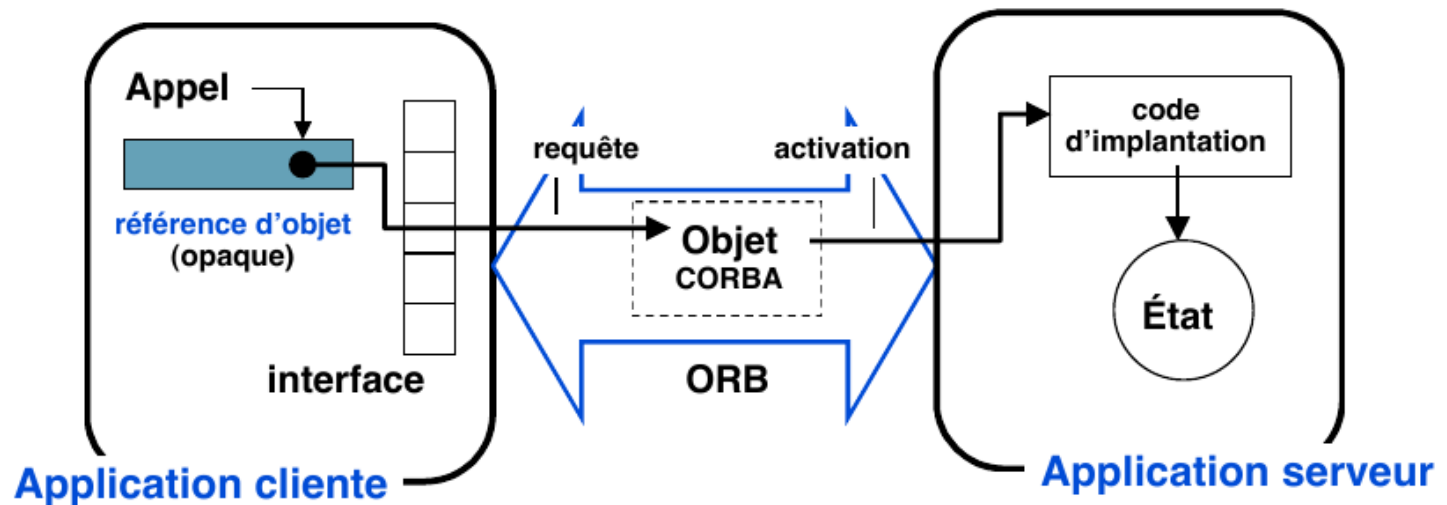


Figure: Krakowiak

# Objet CORBA

- Définition
  - Entité logicielle désignée par une référence, recevant les requêtes émises par les applications clientes
- Objet CORBA
  - Interface IDL
  - Implémentation – *servant*
    - Classe et instance
  - Référence localisant l'objet sur le réseau
    - IOR – Interoperable Reference

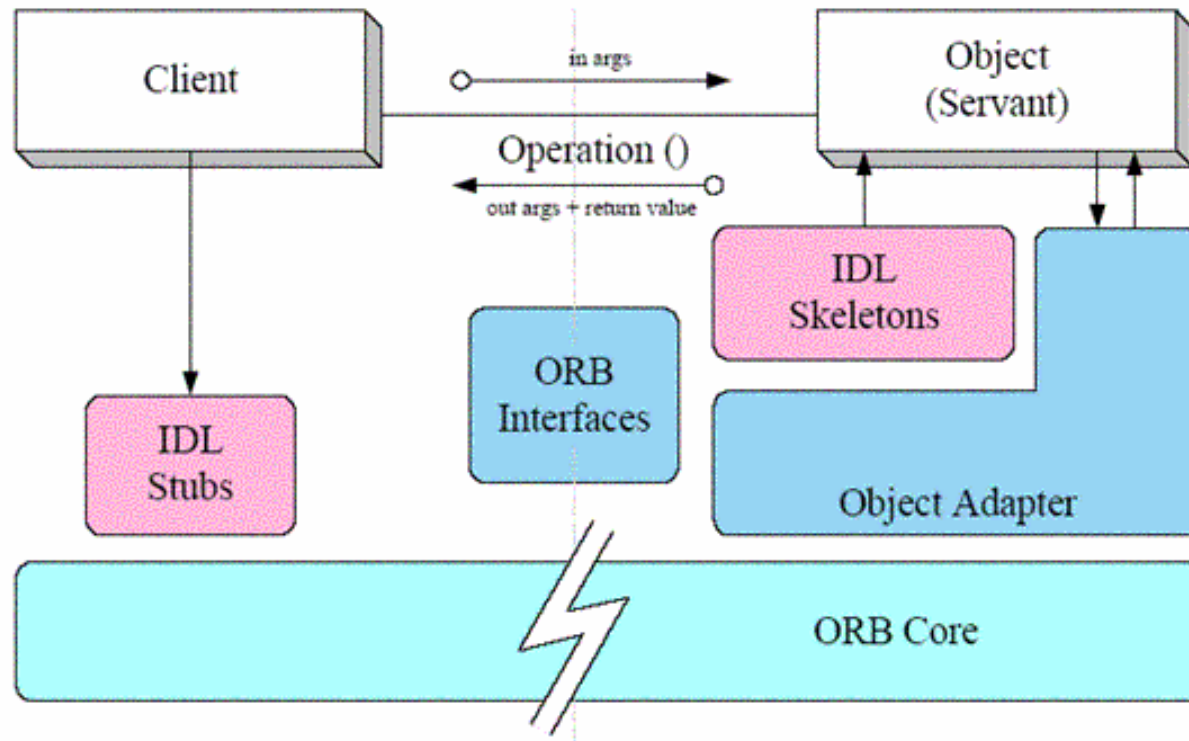
# Protocoles et interopérabilité

- GIOP – General Inter-ORB Protocol
  - Protocole générique de transport pour CORBA
  - Représentation des données commune : CDR
  - Référence des objets : IOR – Inter-Operable Reference
    - Contient : adresse de la machine, version de la couche transport, clef identifiant l'objet (*servant*) dans le serveur
  - Format des messages
- IIOP – Internet Inter-ORB Protocol
  - Instanciation de GIOP sur TCP/IP
  - Interopérable entre implémentations



# Vue d'ensemble

- ORB, stub, skeleton, servant



# OMG IDL – Interface Definition Language

- Langage de description des interfaces
  - Indépendant du langage d'implémentation
  - Purement déclaratif
    - Sépare l'interface et l'implémentation
  - « **Projections** » vers les langages d'implémentation
- Syntaxe proche de C++, Java

# OMG IDL – Exemple

- Exemple d'interface en IDL

```
module Finance
{
  typedef sequence<string> StringSeq;
  struct AccountDetails
  {
    string      name;
    StringSeq   address;
    long        account_number;
    double      current_balance;
  };
  exception insufficientFunds { };
  interface Account
  {
    void deposit(in double amount);
    void withdraw(in double amount) raises(insufficientFunds);
    readonly attribute AccountDetails details;
  };
};
```

# OMG IDL – Elements

- Éléments de base
  - `module` – ensemble de définitions
  - `interface` – interface d'objet, avec héritage
  - `const` - constante
  - `enum` - type énuméré
  - `typedef` – déclaration de type
  - `struct` – type structuré
  - `sequence` – type indexé (tableau)
  - `attribute` – attribut d'interface, éventuellement `readonly`
  -

# OMG IDL – Elements, suite

- Éléments de base
  - Opérations de l'interface
    - Exemple :  
type fonction(in int a, out int b);
    - Sens de passage des paramètres explicite : `in`, `out`, `inout`
  - `exception` – définition d'exceptions
  - Types primitifs
    - `void`, `short`, `long`, `long long`, `float`, `double`, `long double`, `boolean`, `octet`, `string`
    - Projetés vers les types primitifs du langage

# Stubs, squelettes

- Génération automatique de stub à partir de l'IDL
  - *Stub* = souche, talon
  - *Skeleton* = squelette

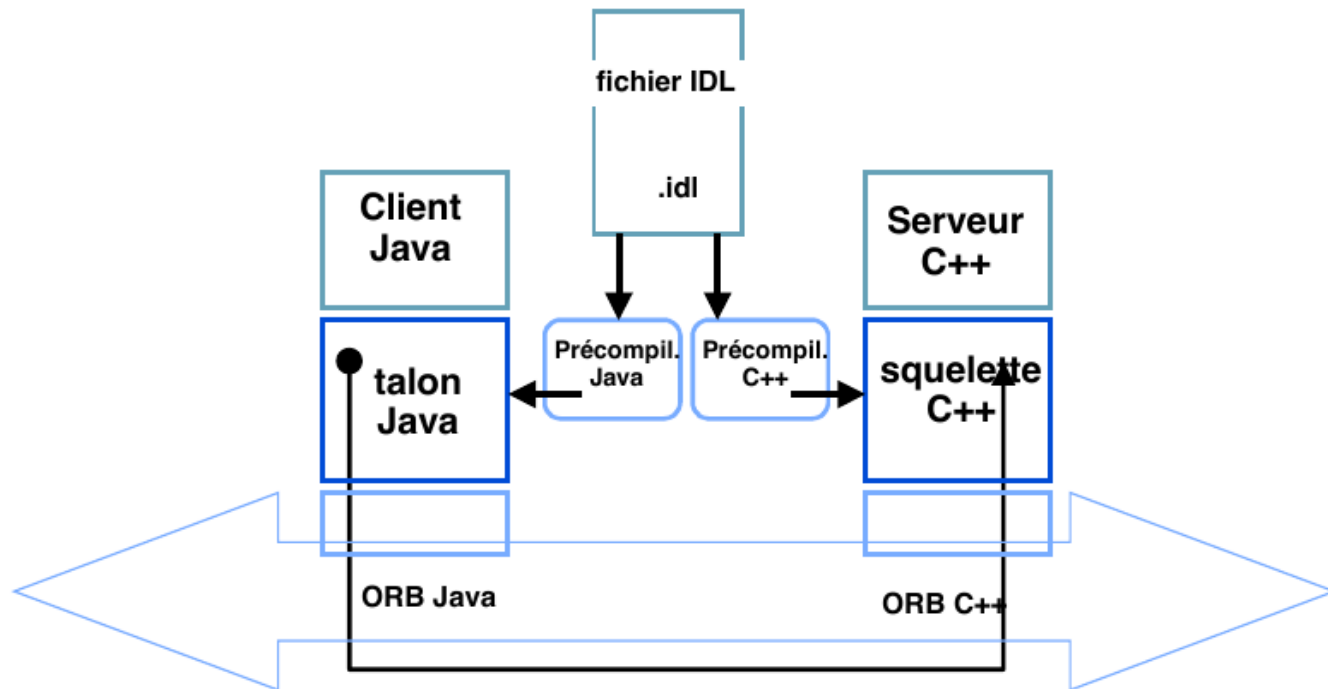
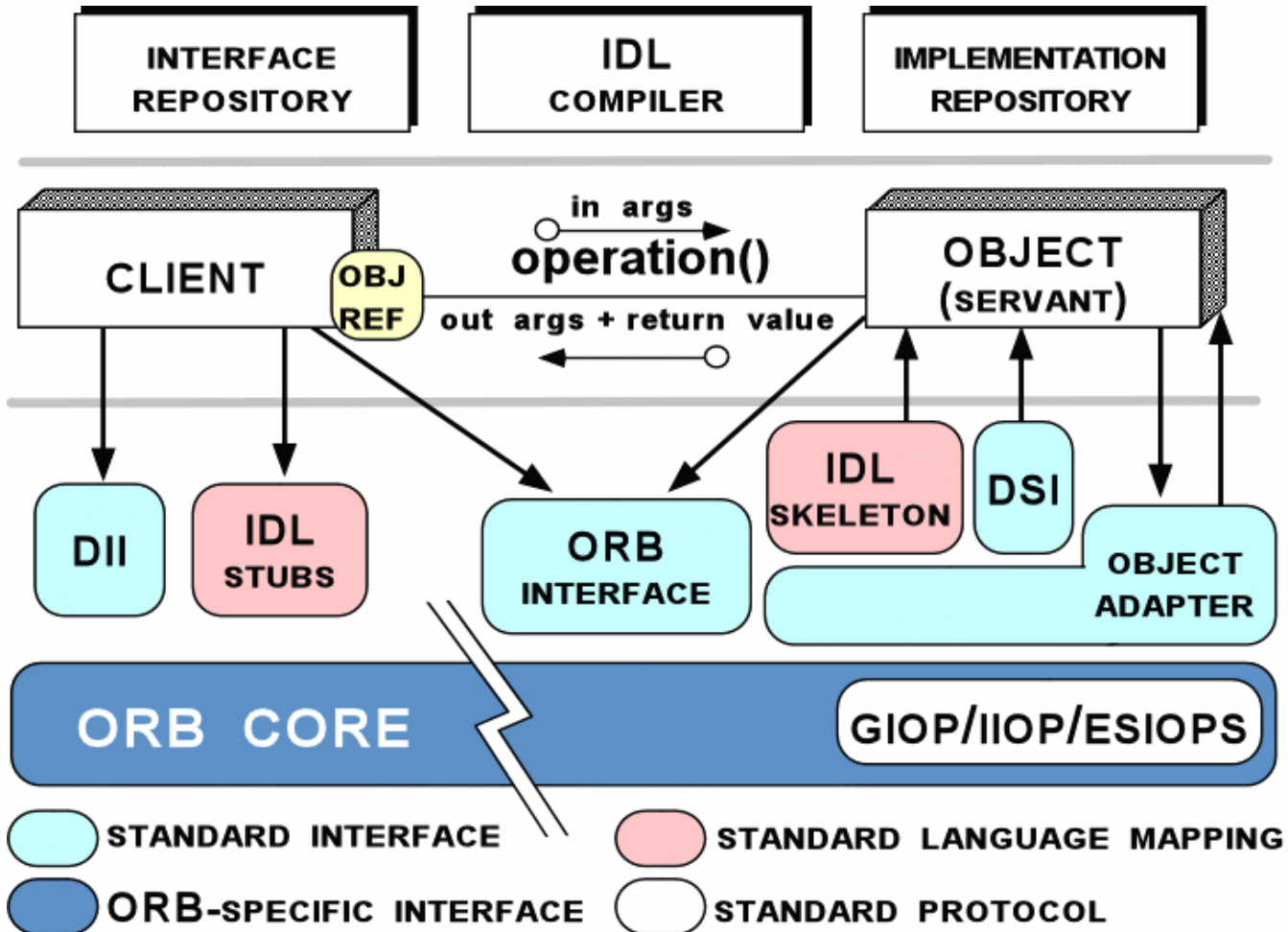


Figure: Krakowiak

# Anatomie de CORBA



# Interface ORB

- L'ORB lui-même est accessible via une référence d'objet
  - Obtenue à l'initialisation
  - `orb = CORBA::ORB_Init(argc, argv);`
  - Paramétré via la ligne de commande
- Fournit des services de base
  - Références initiales vers des objets connus
  - Gestion de l'adaptateur d'objets



# Adaptateur d'objet

- Gère la liaison entre *servant* et ORB ~ démultiplexeur
  - Gère le mapping des objets -> servant
  - Génération des références
  - Adapte les spécificités du langage à l'ORB
  - Durée de vie des objets
  - Activation de processus
  - Modèle de threads
- Implémentations
  - BOA, CORBA 1.0 – spécifique à chaque implémentation
  - POA, CORBA 2.0 – interface standard
    - Plusieurs POA peuvent être instanciés

# Référence

- Désignation unique d'un objet
  - Opaque pour l'utilisateur, pas directement une URL
  - URL possible pour les références initiales
    - corbaloc, corbaname
- Obtenue
  - À la création d'un objet
  - Par l'intermédiaire du serveur de nom (*Naming Service*)
  - Conversion string <-> référence
  - À l'initialisation
    - Référence vers l'ORB, vers le POA, vers le service de nom

# Références IOR, exemples

- IOR
  - Forme binaire
  - Forme « *stringifiée* » (~sérialisée)

```
IOR:00000000000000001749444c3a48656c6c6f4170702f48656c6c6f3a312e30
000000000000100000000000000006c0001020000000000d3134372e3231302e3138
2e310000df7400000021afabcb00000000209db9e72400000001000000000000
000000000004000000000a000000000000010000000100000020000000000001
0001000000020501000100010020000101090000000100010100
```

- En référence initiale : `printf` de l'IOR (ou sauvegarde dans un fichier)
  - Simple et rustique

# Références initiales

- `corbaloc` – URL absolue
  - `corbaloc:iiop:1.2@host1:3075/NameService`
  - Protocole : iiop; version 1.2 ; serveur : host1 ; port : 3075; étiquette : NameService
  - `corbaloc::host1:2809/NameService`
  - Sert principalement à donner la référence du NameService
- `corbaname` – URL dans le NameService
  - `corbaname::foo.bar.com:2809/NameService#x/y`
  - NameService tourne sur foo.bar.com:2809, objet y dans le contexte x
- Ligne de commande standardisée
  - Forme générale `-ORBInitRef nom=URL`
  - `-ORBInitRef NameService=corbaloc::host1:3075/NameService`

# COS – services communs

- CORBA Services – Cos
  - Implémentés sous forme d'objets CORBA
- Service de nommage :
  - Module CosNaming, objet NameService
- Trade service – résolution par type ~ pages jaunes
  - Module CosTrading
- Service d'évènements - EventService
  - Modèle push/pull, couplage faible
- Object Transaction Service – OTS
  - begin, commit, rollback
  - Objet TransactionCurrent

# Service de nommage - NameService

- Annuaire contenant des références d'objets CORBA
  - Association : nom -> IOR
  - Organisé en arboressence, comme un système de fichiers
    - Répertoire -> NamingContext
    - Chemin -> Name
    - Système de nommage nom.extension (id.kind)
- Référence initiale (bootstrap)
  - Opération  
`CORBA::ORB::resolve_initial_references("NameService")`
  - Conversion de `CORBA::Object` vers `CosNaming::NamingContext`
    - Opération CORBA `_narrow`

# Service de nommage – CosNaming.idl

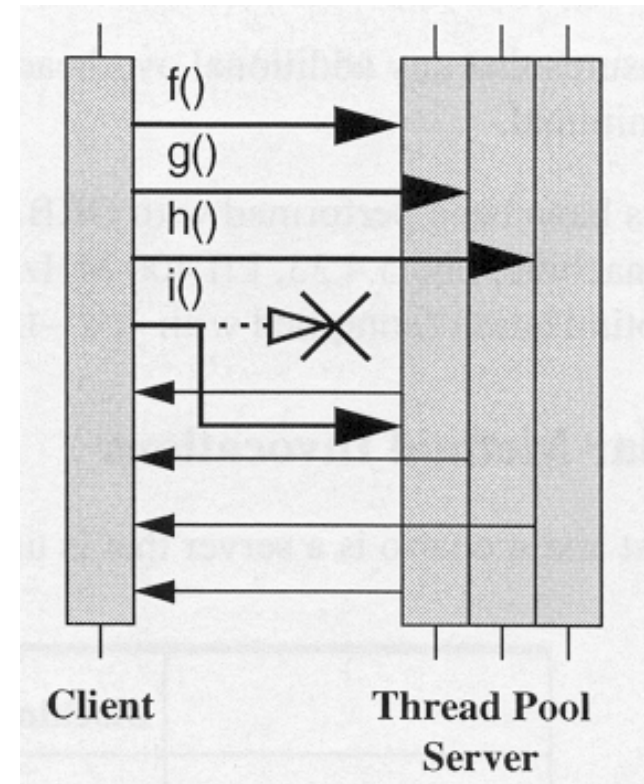
```
module CosNaming {
    typedef string Istring;
    struct NameComponent {
        Istring id;
        Istring kind;
    };
    typedef sequence<NameComponent> Name;
    enum BindingType {nobject, ncontext};
    struct Binding {
        Name binding_name;
        BindingType binding_type;
    };
    typedef sequence <Binding> BindingList;

    interface BindingIterator {
        boolean next_one(out Binding b);
        boolean next_n(in unsigned long how_many,
                      out BindingList bl);
        void destroy();
    };
};
```

```
interface NamingContext {
    void bind(in Name n, in Object obj) raises(...);
    void rebind(in Name n, in Object obj) raises(...);
    void bind_context(in Name n, in NamingContext nc)
        raises(...);
    void rebind_context(in Name n,
                       in NamingContext nc) raises(...);
    Object resolve(in Name n) raises(...);
    void unbind(in Name n) raises(...);
    NamingContext new_context();
    NamingContext bind_new_context(in Name n)
        raises(...);
    void destroy() raises(...);
    void list(in unsigned long how_many,
             out BindingList bl,
             out BindingIterator bi);
};
... // interface NamingContextExt omitted for brevity
};
```

# Modèle d'exécution

- Modèle de thread
  - Dépend de l'implémentation
  - Single thread
    - ex.: MICO
  - Thread par connexion
  - Thread par requête
  - Thread pool
    - ex.: Java
  - Configurable *via* le POA
    - ex.: omniORB, TAO





# Exemple : Hello world !

- Interface Echo.idl

```
interface Echo
{
    string echoString(in string mesg);
};
```

# Exemple : compilation

- Compilation de l'IDL en Java
  - `idlj -fall Echo.idl`
  - Génère les parties client et serveur
  - `EchoOperations.java` – projection de l'interface en Java
  - `EchoStub.java` – souche client
  - `EchoPOA.java` – squelette serveur
  - + divers helpers
- Compilation
  - `javac *.java`

# Exemple : servant

- Implémentation servant en Java : EchoImpl.java
  - Implémentation concrète de l'interface
  - Antisèche (pour les projections de type) : allez voir EchoOperations.java !

```
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;
import java.util.Properties;

class EchoImpl extends EchoPOA
{
    public String echoString(String msg)
    {
        System.out.println("msg: " + msg);
        return msg;
    }
}
```

# Exemple : serveur

```
public class EchoServer
{
    public static void main(String args[])
    {
        ORB orb = ORB.init(args, null); // create and initialize the ORB
        // get reference to rootpoa & activate the POAManager
        org.omg.CORBA.Object objRef = orb.resolve_initial_references("RootPOA");
        POA rootpoa = POAHelper.narrow(objRef);
        rootpoa.the_POAManager().activate();

        // get the naming service
        objRef = orb.resolve_initial_references("NameService");
        NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
        // get object reference from the servant
        EchoImpl echoImpl = new EchoImpl();
        objRef = rootpoa.servant_to_reference(echoImpl);

        // convert the CORBA object reference into Echo reference
        Echo echoRef = EchoHelper.narrow(objRef);

        // bind the object reference in the naming service
        NameComponent path[ ] = ncRef.to_name("echo.echo"); // id.kind
        ncRef.rebind(path, echoRef);

        orb.run(); // start server...
    }
}
```

# Exemple : client

```
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;

public class EchoClient
{
    public static void main(String args[])
    {
        org.omg.CORBA.Object objRef;

        ORB orb = ORB.init(args, null); // create and initialize the ORB

        // get the naming service
        objRef = orb.resolve_initial_references("NameService");
        NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
        // resolve the object reference from the naming service
        objRef = ncRef.resolve_str("echo.echo");

        // convert the CORBA object reference into Echo reference
        Echo echoRef = EchoHelper.narrow(objRef);

        // remote method invocation
        String response = echoRef.echoString("coucou");
        System.out.println(response);
    }
}
```

# Exemple : déploiement

- Déploiement
  - Démarrer le serveur de nom :
    - `tnameserv -ORBInitialPort 2810`
  - Lancer le serveur
    - `java EchoServer -ORBInitRef NameService=corbaloc::host:2810/NameService`
  - Lancer le client
    - `Java EchoClient -ORBInitRef NameService=corbaloc::host:2810/NameService`

# Exemple C++

- Servant

```
#include <CORBA.h>
#include <Naming.hh>
#include "Echo.hh"
using namespace std;

class EchoImpl : public POA_Echo,
                public PortableServer::RefCountServantBase
{
public:

    virtual char* echoString(const char* msg)
    {
        cout << msg << endl;
        return CORBA::string_dup(msg);
    }
};
```

# Exemple C++

- Serveur

```
int main(int argc, char** argv)
{
    CORBA::Object_var objRef;
    // create and initialize the ORB
    CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);
    // get reference to rootpoa & activate the POAManager
    objRef = orb->resolve_initial_references("RootPOA");
    PortableServer::POA_var poaRef = PortableServer::POA::_narrow(objRef);
    poaRef->the_POAManager()->activate();
    // get the naming service
    objRef = orb->resolve_initial_references("NameService");
    CosNaming::NamingContext_var ncRef =
        CosNaming::NamingContext::_narrow(objRef);
    // instantiate the Echo CORBA object
    EchoImpl * echoImpl = new EchoImpl( );
    Echo_var echoRef = echoImpl->_this();
    // bind the object reference in the naming service
    CosNaming::Name name;
    name.length(1);
    name[0].id = (const char*)"echo";
    name[0].kind = (const char*)"echo";
    ncRef->rebind(name, echoRef);
    // start server...
    orb->run();
    return EXIT_SUCCESS;
}
```



# Exemple C++

- Client

```
#include <CORBA.h>
#include <Naming.hh>
#include "Echo.hh"

int main (int argc, char **argv)
{
    CORBA::Object_var objRef;
    // initialize the ORB
    CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);
    // get the naming service
    objRef = orb->resolve_initial_references("NameService");
    CosNaming::NamingContext_var nsRef =
        CosNaming::NamingContext::_narrow(objRef);

    // resolve the "echo" CORBA object from the naming service
    CosNaming::Name name; name.length(1);
    name[0].id = (const char*) "echo";
    name[0].kind = (const char*) "echo";
    objRef = nsRef->resolve(name);
    Echo_var echoRef = Echo::_narrow(objRef);

    // remote method invokation
    cout << echoRef->echoString("coucou") << endl;

    return 0;
}
```

# Compilation C++

- Utilisation de l'implémentation CORBA omniORB
- Compilation IDL
  - `omniidl -bcxx Echo.idl`
  - Génère `EchoSK.cc` et `Echo.hh`
- Serveur de nom
  - `omniNames -start`
- Support Python

# CORBA avancé

- Interface dynamique
- Type Any, DynAny
- Valuetype
- Pointeur intelligent \*\_var
- POA avancé : ServantLocator, ServantActivator
- CosEvent
- Persistance
- Intercepteurs
- Composants
-

# Travail à faire

# Echauffement

- Debuguer les exemples Hello World
  - Il manque quelques import, la gestion des exceptions
- Compiler et faire tourner les exemples Hello World
  - En local
  - A distance
  - Avec le serveur d'un voisin
- Modifier la méthode pour manipuler différents types : string, long, structures
  - Observez les changements dans EchoOperations.java

# Application bancaire

- Nous allons simuler le fonctionnement d'une application bancaire
- Deux types d'objets :
  - Banque : **Bank**
  - Compte bancaire : **Account**
- Respectez les interfaces IDL données pour pouvoir tester avec les implémentations des autres

# Application bancaire

- Implémentez l'interface Account
  - Implémentez le serveur
  - Écrivez un client de test

```
interface Account
{
    void deposit(in unsigned long amount );
    void withdraw(in unsigned long amount );
    long balance();
};
```

# Application bancaire

- Implémentez l'interface Bank
  - Implémentez le serveur
  - Écrivez un client de test
- Il s'agit du pattern célèbre *Factory* !
  - Un objet qui sert à créer d'autres objets

```
interface Bank
{
    Account create ();
    void destroy(in Account a);
};
```



# Application bancaire

- On ajoute le virement inter-bancaire
  - Pourquoi cette opération est-elle dans l'interface Bank ?
- Le serveur devient à son tour client
  - Bank et Account sont des références, pas des copies d'objet

```
interface Bank
{
    Account create ();
    void destroy(in Account a);
    Account move(in Bank target, in Account a);
};
```

# Pour les plus téméraires

- Proposer une version Java et une version C++ de l'application bancaire
  - Vérifier l'interopérabilité
- Proposer une version CORBA du calcul distribué de Pi du TP Java RMI
  - Quel est l'intérêt du CORBA par rapport à RMI dans ce cas ?

À vous de jouer !

*inria*  
informatics mathematics

<http://dept-info.labri.fr/~denis/>